A Geometric Approach to Obtain a Bird's Eye View From an Image

Ammar Abbas *
VGG, Dept. of Engineering Science
University of Oxford
United Kingdom
sabbas.ammar@gmail.com

Andrew Zisserman
VGG, Dept. of Engineering Science
University of Oxford
United Kingdom
az@robots.ox.ac.uk

Abstract

The objective of this paper is to rectify any monocular image by computing a homography matrix that transforms it to a geometrically correct bird's eye (overhead) view.

We make the following contributions: (i) we show that the homography matrix can be parameterised with only four geometric parameters that specify the horizon line and the vertical vanishing point, or only two if the field of view or focal length is known; (ii) We introduce a novel representation for the geometry of a line or point (which can be at infinity) that is suitable for regression with a convolutional neural network (CNN); (iii) We introduce a large synthetic image dataset with ground truth for the orthogonal vanishing points, that can be used for training a CNN to predict these geometric entities; and finally (iv) We achieve state-of-the-art results on horizon detection, with 74.52% AUC on the Horizon Lines in the Wild *dataset. Our method is fast and robust, and can be used to remove perspective distortion from videos in real time.*

1. Introduction

Understanding the 3D layout of a scene from a single perspective image is one of the fundamental problems in computer vision. Generating a bird's eye (or overhead, or orthographic) view of the scene plays a part in this understanding as it allows the perspective distortion of the ground plane to be removed. This *rectification* of the ground plane allows the scene geometry on the ground plane to be measured directly from an image. It can be used as a preprocessing step for many other computer vision tasks like object detection [19, 29] and tracking [10], and has applications in video surveillance and traffic control. For example, in crowd counting, where perspective distortion affects the crowd density in the image, the crowd density can instead be predicted in the world [24].

*The author is now at Latent Logic, Oxford

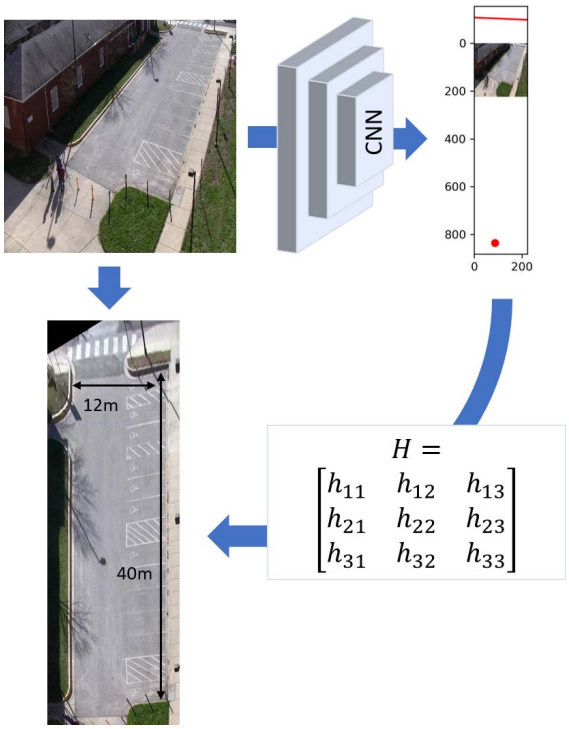

Figure 1: An overview of our method for obtaining the bird's eye view of a scene from a single perspective image. A CNN is used to estimate the vertical vanishing point and ground plane vanishing line (horizon) in the image, as shown by the red dot and line in the example. This point and line in turn determine the homography matrix, H, that maps the image to the overhead view with perspective distortion removed. Measurements on the ground plane (up to an overall scale) can then be made directly on the rectified image.

Since obtaining a bird's eye view from an image involves computing a rectifying planar homography, it might be thought that the most direct way to obtain this transformation would be to regress the eight parameters that specify the homography matrix. Instead, we show that this homography can be parametrised with only four parameters corresponding to the vertical vanishing point and ground plane

vanishing line (horizon) in the image, and that these geometric entities can be regressed directly using a Convolutional Neural Network (CNN). Furthermore if the focal length of the camera is known (or equivalently the field of view) from the EXIF data of the image, then only two further parameters are required (corresponding to the vanishing line). We show that given these minimal parameters, the homography matrix that transforms the source image into the desired bird's eye view can be composed through a sequence of simple transformations. Furthermore, the geometric entities can also be used directly for scene understanding [16].

For the purpose of training a CNN, we introduce and release [1] the largest up-to date dataset which contains the ground truth values for all the three orthogonal vanishing points with the corresponding internal camera matrices, and tilt and roll of the camera for each image. We also propose a novel representation for the geometry of vanishing lines and points in image space, which handles the standard challenge that these entities can lie within the image but also can be very distant from the image, making this representation a good choice for the network prediction.

In summary, we make the following four contributions: (i) we propose a minimal parametrisation for the homography that maps to the bird's eye view. This requires only four parameters to be specified (the vanishing point and vanishing line), or only two if the focal length of the camera is known (the vanishing line); (ii) we propose a new geometric representation for encoding vanishing points and lines that is suitable for neural network computation; (iii) we generate and release a large synthetic dataset, CARLA-VP, that can be used for training a CNN model to predict vanishing points and lines from a single image; and (iv) we show that a CNN trained using our four scalar parameterisation exceeds the performance of the state of the art on standard real image benchmarks for horizon detection [36].

We also show that current methods [18, 22] can fail for horizon prediction when the actual horizon line lies outside of the image. This failure is due to the parameterization used, as well as to the training data used (which mostly contains horizon lines inside the image since it is easier to annotate them). We avoid this annotation problem by using synthetic data for training, where images can be generated following any desired distribution and the annotations are more precise as well. We compare to results on a benchmark dataset [30] in section 6.4.

2. Related Work

Estimating homographies: Bruls et al. [7] use GANs to estimate the bird's eye view; however, since they don't enforce a pixel-wise loss, the geometry of the scene may not be correctly recovered as they mention in failure cases. Moreover, they train and test only on first person car driver views [26] where some assumptions can be made (pitch≈0, roll≈0). Liu et al. [23] pass an additional relative pose to a CNN for view synthesis which contains information about the relative 3D rotation, translation and camera internal parameters.

Estimating the focal length of the camera: One of the ways to calculate focal length is by estimating the field of view from the image. The focal length f is inversely related to the field of view γ of the camera given constant image width w as:

$$\tan(\frac{\gamma}{2}) = \frac{\frac{w}{2}}{f} \tag{1}$$

Workman et al. [35] use this approach to predict a camera's focal length by estimating the field of view directly from an image using a CNN. However, since they only predict horizontal field of view, they assume that the camera has equal focal length on both the axes which may not be true. In addition, based on the findings in [20], we know that predicting the field of view directly from an image can be a challenging task since two similar looking images may have large differences in field of view. We estimate the focal length of the camera from the horizon line and the vertical vanishing point (and describe the advantages in section 6.3).

Computing vanishing points and lines: One simple way to estimate the horizon line or the vertical vanishing point is by finding the intersection point of the lines in the image which belong to the orthogonal directions in the image. More specifically, this could involve using a Hough transform on the detected lines to vote among the candidate vanishing points [34], and many other voting schemes have been investigated [9], including weighted voting [31] and expectation maximization [3]. More recently, Lezama et al. [22] vote both in the image domain and PClines dual spaces [14]. The above methods have a limitation as they rely on line detection as the core step and may fail when the image does not have lines in the major directions. Fortunately, there are other important cues in an image which help us to estimate the horizon line or the vanishing points such as colour shifts, distortion in objects' shapes, change in texture density or size of objects around the image *etc*. and that is where deep learning methods can help us.

Datasets: There are a few existing datasets which contain the ground truth for the three orthogonal vanishing points in the scene namely, *Eurasian Cities* dataset [5], *York Urban* dataset [12] and the *Toulouse Vanishing Points* dataset [2]. However, these datasets contain only around 100 images in total. Borji [6] propose a CNN based method which is trained by annotating vanishing points in YouTube frames. Recently, Workman et al. [36] collected a new dataset called

[1] https://drive.google.com/open?id=1o9ydKCnh0oyIMFAw7oNxQohFa0XM4V-g

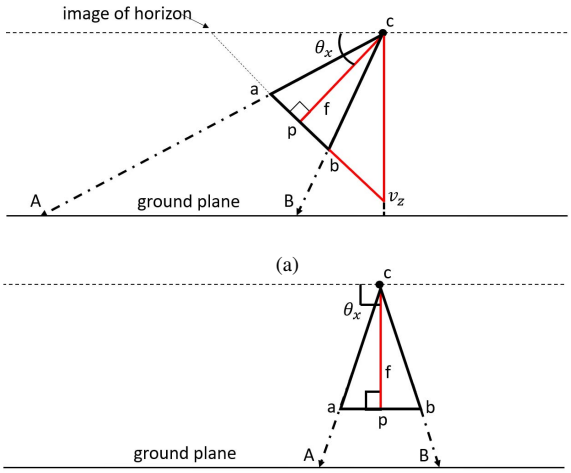

Figure 2: Side view of a camera viewing a scene. (a) The camera is tilted with an angle θ_x. Its centre is represented by c and f is the focal length. $\overline{ab}$ is the image plane, p the principal point, and v_z the vertical vanishing point. (b) The camera is rotated to look directly vertically down on the scene – the bird's eye view.

Horizon Lines in the Wild (HLW) which contains around 100K images with ground truth for the horizon line. However, their dataset mostly contains images where the horizon line lies within the image, and does not contain explicit labeling for the orthogonal vanishing points. Because of the unavailability of a large dataset which contains the orthogonal vanishing points, we generate a large-scale synthetic dataset that contains the required ground truth. This allows us to train a CNN to predict these geometric entities. We discuss this in detail in section 5

3. Predicting a homography from the horizon line and the vertical vanishing point

In the following we assume that we know the vertical vanishing point and horizon line in the image, and show geometrically how these are used to compute the rectifying homography matrix. In section 4 we describe how to estimate these geometric entities using a CNN.

The method involves applying a sequence of projective transformations to the image that are equivalent to rotating the camera and translating the image in order to obtain the desired bird's eye view. As shown in figure 2 the key step is to use the horizon line to determine the rotation required, but in order to know the rotation angle from the horizon we require the camera internal calibration matrix. Assuming that the camera has square pixels (zero skew) and that the principal point is at the centre of the image, then the only unknown parameter of the internal calibration matrix is the focal length, and this can be determined once both the vertical vanishing point and horizon are known as described below.

Preliminaries. We will use the following relationship [17] between image coordinates before and after a rotation of the camera about its centre:

$$\mathbf{x}' = \mathtt{K}\mathtt{R}\mathtt{K}^{-1}\mathbf{x} \tag{2}$$

where $\mathbf{x}$ represents image pixels for scene coordinates $\mathbf{X}$ before the camera rotation, and $\mathbf{x}'$ are the resultant image pixels for the same scene coordinates $\mathbf{X}$ after the rotation, and the internal calibration matrix $\mathtt{K}$ is given by

$$\mathtt{K} = \begin{bmatrix} f & 0 & w/2 \\ 0 & f & h/2 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

where f is the focal length of the camera, w is the width of the image, and h the height of the image.

To compute the matrix $\mathtt{K}$, we only need to find the focal length f of the camera. As explained in [17] the focal length can be obtained directly from the relationship

$$\mathbf{h} = \boldsymbol{\omega}\mathbf{v} \tag{4}$$

where $\mathbf{h}$ is the horizon line and $\mathbf{v}$ the vertical vanishing point, and $\boldsymbol{\omega}$ is known as the *image of absolute conic* which is unaffected by the camera rotation and is given by $\boldsymbol{\omega} = (\mathtt{K}\mathtt{K}^T)^{-1}$.

The rotation matrix $\mathtt{R}$ in equation (2) can be composed of rotations about different axes. We use this property to first rotate the camera about its principal axes to correct for the roll of the camera, and then about the x-axis of the camera to reach an overhead view of the scene. We next describe the sequence of projective transformations.

Step A: removing camera roll. The first step is to apply a rotation about the principal axis to remove any camera roll, so that the camera's x-axis is parallel to the X-axis of the world. See step A in figure 3 for its effect. The roll of the camera is computed from the horizon line. Given a horizon line of the form $ax + by + c = 0$, the roll of the camera θ_z is given by $\theta_z = \tan^{-1}(\frac{-a}{b})$. The rotation matrix R_{roll} for rotating about the principal axis is computed using θ_z.

Step B: removing camera tilt. The next step is to rotate about the camera x-axis to remove the camera tilt. See step B in figure 3 for its effect. The rotation matrix for rotation about the camera x-axis requires only one parameter which is the camera tilt θ_x. The camera tilt can be found from the focal length and one of the geometrical entities, either the horizon line or the vertical vanishing point. Given the focal length of the camera f and the perpendicular distance from the vertical vanishing point to the principal point $\|v_z\|$, we can find tilt θ_x of the camera as $\theta_x = \frac{\pi}{2} - \tan^{-1}(\frac{\|v_z\|}{f})$. See figure 2 for the corresponding notation. At this point, the homography matrix $\mathtt{H}_{rot}$ is given as:

$$\mathtt{H}_{rot} = \mathtt{K}\mathtt{R}_{tilt}\mathtt{K}^{-1}\mathtt{R}_{roll} \tag{5}$$

Original Image Step A Step B Step C Step D

Figure 3: Step-by-step transformations of the first image to obtain the desired bird's eye view. The different sub-figures correspond to images obtained after performing the steps described in section 3 for predicting the homography matrix.

where $\mathtt{R}_{tilt}$ is the rotation matrix for rotating about the x-axis to recover the camera tilt.

Step C: image translation. Once we have the effect of camera rotation, we also need to translate the camera so that it is directly above the scene and captures the desired bird's eye view. We achieve this by applying H_{rot} to the four corners of the source image which returns the corresponding corners for the destination image. We define a translation matrix which maps the returned corners to the corners of our final canvas, thereby giving us the full view of the scene from above. See step C in figure 3.

Step D: optional rotation. We also have an optional step which can be seen in step D in figure 3. It deals with aligning the major directions in the image with the axes in the Cartesian coordinate system by rotating the final image by an angle θ_{align}. This angle can be obtained from one of the principal horizontal vanishing points that relates to one of the major directions in the image. We show in section 4.1 how to represent this vanishing point by a single scalar.
In summary, the steps of the algorithm are:

- Calculate the focal length of the camera using the predicted horizon line and the vertical vanishing point from a single image.
- Calculate the camera roll from the horizon line which gives us $\mathtt{R}_{roll}$.
- Calculate the camera tilt from the focal length and the vertical vanishing point which in turn is used to calculate R_{tilt}
- Calculate the translation matrix T_{scene} using the homography matrix H_{rot} from eq. 5 to map the corners of the image to the destination image.
- *(Optional)* Calculate R_{align} from the principal horizontal vanishing point in the scene.
- Compose all above transformation matrices together to calculate the final homography matrix which is given as follows:

$$\mathtt{H} = \mathtt{R}_{align}\mathtt{T}_{scene}\mathtt{K}\mathtt{R}_{tilt}\mathtt{K}^{-1}\mathtt{R}_{roll} \tag{6}$$

4. Predicting the horizon line and the vertical vanishing point

In this section we describe how the geometric entities are represented in a form suitable for regression with a CNN. The key point is that the entities can be at infinity in the image plane (e.g. if the camera is facing down then the vanishing line is at infinity) and so a representation is needed to map these entities to finite points for the CNN prediction. To achieve this we borrow ideas from the standard stereographic projection used to obtain a map of the earth [32].

4.1. Representing the geometry of the horizon line and the vanishing points

We first describe the representation method for a point. See figure 4 for the notation introduced ahead. Suppose there is a sphere of radius r which is located at point $(0, 0, r)$, and let the image plane be at $z = 0$. Then we can draw a line connecting any point P on the image plane to the sphere centre. The point s on sphere where this line intersects the sphere is given by $s = r\frac{\boldsymbol{v}}{\|\boldsymbol{v}\|}$ where v is a vector from the sphere centre to P and s is a 3-D point on sphere. Finally, we project the point s onto the image plane at Q using orthogonal projection. This effectively allows us to represent any 2D point P on the image by a point Q in a finite domain (within a circle of radius r), irrespective of whether the original point P is finite or at infinity.

For a line l, we take a slightly different approach to represent its geometry. We draw a plane which connects the line l to the centre of the sphere. There is a one-to-one mapping between the line l and the plane drawn corresponding to it. The normal n to the plane from the sphere centre intersects the surface of the sphere in the lower hemisphere at a point s. Once again, we orthogonally project this point s onto the plane. This gives a unique finite point representation for any line l in the infinite plane. In this way, we can represent the horizon line and the vertical vanishing point in the image by a total of four scalars which lie in the range $[-r, r]$.

The optional principal horizontal vanishing point can be represented by a single scalar. We know that the horizontal vanishing points lie on the horizon line, so we only need to

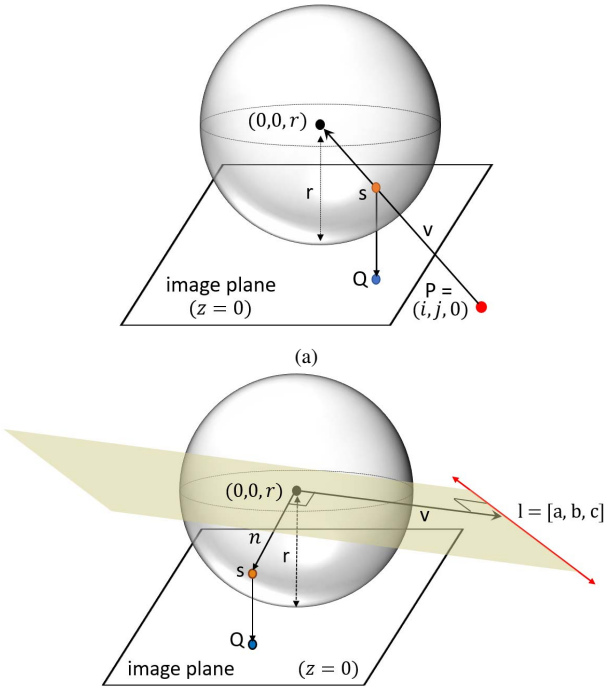

Figure 4: Representation of the geometry of a vanishing point and a horizon line. (a) A point P (which could be a point at infinity) is represented by a finite point Q on the 2D plane. As the point Q is projected from the intersection point s on the lower hemisphere, it is constrained to lie within a circle of radius r on the image plane. (b) A line l is represented by a finite point Q on the image plane. The plane connects the centre of the sphere with the line l. Point s is the intersection of the plane normal with the sphere boundary, and Q is the projection of s on the image plane.

Dataset	Training	Validation	Test
HLW [36]	100553	525	2018
VIRAT Video [30]	-	-	11 videos
CARLA-VP	12000	1000	1000

Table 1: Comparison of the number of examples in training, validation, and test set for different datasets. Note: The videos for VIRAT dataset aren't divided into different training, validation or test sets by the publishers.

measure its position on the horizon line. We do so by measuring the angle between two vectors: a vector $\mathbf{v_1}$ which goes from the sphere centre C to the required horizontal vanishing point and another vector $\mathbf{v_2}$ which is normal to the horizon from C.

5. The CARLA-VP Dataset

There is no large scale dataset with ground truth for the horizon line and the vertical vanishing point available for training a CNN, so here we generate a synthetic training dataset. Table 1 gives statistics on relevant datasets.

5.1. Synthetic dataset

We use CARLA [13] which is an open-source simulator built over the Unreal Engine 4 to create our dataset. It generates photo-realistic images with varying focal length, roll, tilt and height of the camera in various environments.

We choose a uniformly random value for the height of the camera ranging from a ground person's height to around 20 metres. We also choose a uniformly random value for tilt of the camera in the range $(0°, 40°]$. We choose a value for camera roll from a normal distribution with $\mu = 0°$ and $\sigma = 5°$ which is truncated in the range $[-30°, 30°]$.

CARLA provides the ability to change the field of view of the camera. This allows us to effectively change the focal length of the camera as given in equation (1). We use a uniformly random value for field of view from the range $[15°, 115°]$ which is carefully selected based on the images that are generally captured or are obtained from traffic surveillance cameras. The different parameters that we have discussed above allow us to generate a wide variety of images with different aspect ratios that resemble real-world images. We will refer to this dataset as *CARLA-VP* (i.e. CARLA with Vanishing Points). See figure 5 for a few samples from the dataset.

5.2. Ground Truth Generation

Synthetic datasets allow us to create precise ground truths. We mentioned above that we can change tilt, roll or yaw of the camera in the CARLA simulator. This gives us the value for the camera's rotation matrix $\mathtt{R} = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 \end{bmatrix}$ by composing it as a composition of individual rotation matrices. Similarly, we also know the internal calibration matrix $\mathtt{K}$ of the camera as CARLA uses a simplified form and we already know the focal length (1).

Using $\mathtt{K}$ and $\mathtt{R}$, we can generate ground truth for the orthogonal vanishing points. Consider a point at infinity in the z direction, $\mathbf{z}_\infty$, which is represented as $\begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}^T$ in homogeneous coordinates, and its image v_z. Then by the camera's projection equation, we have:

$$\mathbf{v}_z = \mathtt{K} \left[\begin{array}{c|c} \mathtt{R} & \mathbf{t} \end{array} \right] \mathbf{z}_\infty = \mathtt{K} \left[\begin{array}{ccc|c} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \end{array} \right] \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\mathbf{v}_z = \mathtt{K}\mathbf{r}_3 \tag{7}$$

Similarly, we can also solve for the orthogonal horizontal vanishing points in the scene which are given by $\mathbf{v}_x = \mathtt{K}\mathbf{r}_1$ and $\mathbf{v}_y = \mathtt{K}\mathbf{r}_2$, and the horizon line is given by $\mathbf{h} = \mathbf{v}_x \times \mathbf{v}_y$.

6. Experiments

In this section, we perform a range of experiments to evaluate our method both qualitatively as well as quantita-

Figure 5: Sample images from the CARLA-VP dataset. The images show a wide variety of settings i.e. different camera positions and orientations, different weathers and different times of the day.

tively. We first explain the performance measures and conduct an ablation study of the method in section 6.3, where we also compare different CNN architectures. We then evaluate our method on videos and compare its performance quantitatively on the VIRAT Video dataset with some qualitative results on the real-world images. Finally, we compare our horizon detection method with previous state-of-the-art methods.

6.1. Performance Measures

We use two performance measures. The first is the area under the curve (AUC) metric proposed by Barinova et al. [5] for evaluating horizon line estimation. For each test image sample, the maximum difference between the height of the ground truth and estimated horizon over the image, divided by the image height, is computed; and these values are then plotted for the test set, where the x-axis represents the error percentage and the y-axis represents the percentage of images having error less than the threshold on the x-axis. The AUC is measured on this graph.

The second performance measure evaluates the camera internal and external parameters, in particular the field of view (which depends on the predicted focal length), and the roll and tilt of the camera. We measure the error in these parameters in degrees. Note, these quantities are not directly estimated by the CNN, but are computed from the predicted vertical vanishing point and horizon line.

6.2. Implementation details

The final layer of the network is required to predict four scalars, and this is implemented using regression-by-classification as a multi-way softmax for each scalar over b discretization bins. The number of discretization bins is chosen as $b = 500$ in our experiments. An alternative would be to directly regress each scalar using methods similar to [15, 27], but we did not pursue that here.

At test time, we consider the c bins with the highest probability, and use a weighted average of these bins by their probabilities to calculate the regressed value. We find that $c = 11$ gives the best performance on the validation set.

Model Parameterization	Field of view	Camera tilt	Camera roll
Horizon and field of view	6.061°	2.663°	1.238°
Horizon and vertical vanishing point	**4.911°**	**2.091°**	**0.981°**

Table 2: Comparison of the error in estimated internal and external camera parameters on the CARLA-VP dataset using different parameterization techniques (lower is better). It can be seen that the CNN trained to output the horizon line and the vertical vanishing point gives better performance.

CNN Architectures	Field of view	Camera tilt	Camera roll
VGG-M	6.163°	2.332°	1.534°
VGG-16	5.385°	1.887°	1.207°
Resnet-50	4.509°	1.755°	1.104°
Resnet-101	4.534°	1.652°	1.234°
Inception-V1	6.773°	2.374°	1.456°
Inception-V4	**4.130°**	**1.509°**	**0.853°**

Table 3: Comparison of the error in estimated internal and external camera parameters on the CARLA-VP dataset using different CNN architectures as a component of our pipeline (lower is better).

The CNN is trained using TensorFlow [1] v-1.8 in Python 3.6. It is initialized with pre-trained weights from ImageNet classification [11]. All layers are fine-tuned as the task at hand is inherently different from the image classification task. We use the Adam optimizer [21] with default parameters. The training starts with an initial learning rate of 1e-2 which is divided by 10 up-to 1e-4 whenever the validation loss increases.

6.3. Ablation Study

Field of view vs vertical vanishing point. We discussed in section 4 that our method for calculating the bird's eye view involves estimating the internal and external parameters of the camera. We do this by estimating the horizon line and the vertical vanishing point from a given image. This involves predicting four different scalars. However, we can further reduce the number of parameters by predicting the field of view instead of the vertical vanishing point. This is an even more compact representation which uses only three scalars. It allows us to calculate the focal length directly from the field of view as in (1) [20], and the tilt and roll of the camera from the horizon line and focal length.

We evaluate this approach to see how it performs against our original method. The results are shown in table 2. We observe that the four scalar parameterization does better in estimating all the internal and external parameters of the camera. We believe that one of the major reasons is that the vertical vanishing point is easier to estimate given that the orientation of the ground plane or the direction of vertical

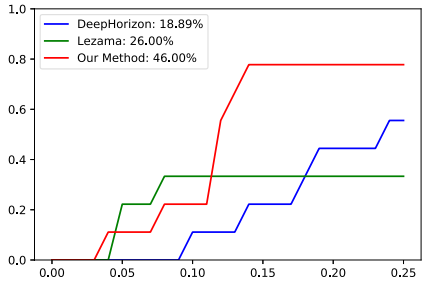

Figure 6: Horizon line detection AUC performance on the VIRAT Video dataset. Comparison of our method (trained on synthetic data) against DeepHorizon [36] and Lezama [22]. The dataset contains a variety of images with various positions and orientations of the horizon line. Our method does significantly better than the state-of-the-art.

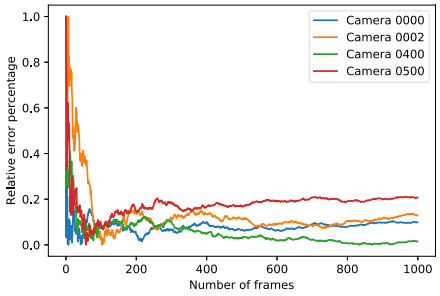

Figure 7: Reduction of relative error in estimating the focal length of the camera as the estimates from more frames are averaged for different videos in the VIRAT Video dataset. We observe that the estimate asymptotes at around 400 frames.

lines on the ground plane is directly observable from the image. On the other hand, the camera's field of view can be difficult to estimate given the fact that two images which are captured from cameras with different focal lengths and different distances to the objects may appear very similar.

There are other advantages of our method as well. It is easier to verify the vertical vanishing point manually from an image. It also gives us an additional method for calculating the tilt of the camera and we can average it with the tilt value calculated from the horizon line. Furthermore, the focal length of the camera is relatively more sensitive to small errors at large values of the field of view due to the $\tan$ relation in (1) (the focal length is inversely proportional to $\tan$ of the field of view. Therefore, for large values of the field of view, a small change in the field of view (e.g. change from 115 to 117 compared to 45 to 47) will cause f to change more since the slope of the tangent increases very quickly as it approaches $\pi/2$).

Choice of trunk architecture. We compare the performance using a number of different and popular CNN architectures. In each case, the CNN is initialized by pre-training on ImageNet classification. We start with a simple model *i.e.* VGG-M [8] with relatively few parameters, and then train progressively more complex and deeper CNNs. Table 3 shows the comparison of the tested networks on the CARLA-VP dataset. We use the best performing Inception-v4 [33] architecture for the remaining results.

6.4. Comparison with other methods

We compare our method for estimating the horizon line on two public image dataset benchmarks.

6.4.1 Comparison on the VIRAT Video dataset

The VIRAT video dataset [30]. This dataset contains videos with fixed cameras (table 1) along with the corresponding homography matrices for the ground planes. It also contains object and event annotations. We use single images extracted from videos in this dataset and extract the ground truth horizon lines from the given homography matrices using (7).

We compare our method, trained on the synthetic CARLA-VP dataset, to two other methods: DeepHorizon [36] using the provided API; and Lezama [22] using the code published by the authors. Therefore, this dataset is unseen for all three methods. See figure 6 for the results.

We observe that our method outperforms DeepHorizon (state-of-the-art) and Lezama by a significant margin. Upon closer inspection, we see that the DeepHorizon method struggles on images where the horizon line lies outside the image, while our method is able to do well on such images. One of the reasons could be that DeepHorizon gives good weightage to segmentation between the ground plane and the sky to aid the horizon prediction, but this cue may not be available when the camera is titled significantly.

We show qualitative results for some of the scenes from the VIRAT Video dataset in figure 8, which contains the original images and their corresponding bird's eye views. The obtained bird's eye views have the correct geometric proportions for different objects present in the scene such as dimensions of lane markings and roads. This means that we can obtain Euclidean measurements in the scene if we know one reference distance in the image. We observe that our method is able to transfer well to the real-world images and generates veridical views.

Real time performance on Videos. Since our method does not involve any other refinement steps like expectation maximization *etc.* as used in [37], it is very fast and takes around 40 ms per image on a lower-middle end GPU (GTX 1050 Ti). This amounts to 25 frames per second, thus making it suitable for application to videos in real time.

Here, we evaluate a simple approach which can be used to improve the performance. We apply our method to different videos from the VIRAT Video dataset and average the values for the internal and external parameters of the cam-

Method	Datasets	Post-Processing	AUC
Lezama et al. [22]	*(requires no training)*	✓	52.59%
Zhai et al. [37]	110K Google Street	✓	58.24%
Workman et al. [36]	HLW+500K Google Street	✗	69.97%
Workman et al. [36]	HLW+500K Google Street	✓	71.16%
Ours	HLW	✗	**74.52%**

Table 4: Horizon line detection AUC performance on the HLW test dataset. Comparison of our method against other horizon-line detection methods. The datasets column shows the datasets the methods were trained on.

Figure 8: (Left) Source images (Right) Bird's eye view of the corresponding source images which are automatically calculated using our method.

era (rather than the homography matrix). This allows us to refine our estimated values continuously and get more reliable and stable results. We observe that the estimate of the camera parameters gets more accurate as more frames are averaged from the video. See figure 7 for a visualization of the focal length error. The estimated value for the focal length approaches the ground truth value as the number of frames increases.

6.4.2 Comparison on the HLW Dataset

In this section, we present our results on the latest horizon detection dataset known as *Horizon Lines in the Wild* (HLW).

The Horizon Lines in the Wild (HLW) dataset [36]. This dataset contains around 100K images with ground truths for the horizon line. The dataset mostly contains images with a very small tilt or roll of the camera and the camera is close to a ground person height. This causes the horizon line to be visible in most of the images.

We use pre-initialized weights from ImageNet to train our method on the training set of the HLW dataset to compare with other methods. See table 4 for a summary of performance of different methods on the HLW test set. We achieve 74.52% AUC, outperforming the previous state-of-the-art method Workman et al. [36] with a relative improvement of 4.72%.

Our network predicts the geometry in one forward pass, without any kind of post-processing involved. Compared to this, Lezama et al. [22] detect line segments in the image initially, and compute vanishing points from them which gives the horizon line. Zhai et al. [37] estimates horizon line candidates from the CNN. Then they estimate the zenith vanishing point using these horizon lines. Based on this, they estimate the horizontal vanishing points on the horizon line candidates and select the horizon line with maximum score. Workman et al. [36] estimate the horizon line directly from the image using a CNN, but they use further post-processing techniques to achieve their best results.

7. Conclusion

We have presented a complete pipeline for removing perspective distortion from an image, and obtaining the bird's eye view from a monocular image automatically under geometric constraints. Our method can be used as plug and play to help other networks which suffer from multiple-scales due to perspective distortion such as vehicle tracking [28], crowd counting [24, 25] or penguin counting [4] *etc*. Our method is fast, robust and can be used in real-time on videos to generate a bird's eye view of the scene.

Note, the finite points used to represent the geometric entities in this work do not correspond directly to observable features in the image. A possible improvement in future work would be to design a projection method so that they do correspond.

Acknowledgements: We would like to thank Nathan Jacobs for his help in sharing the DeepFocal [35] dataset. We are grateful for financial support from the EPSRC Programme Grant Seebibyte EP/M013774/1.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[2] V. Angladon, S. Gasparini, and V. Charvillat, "The toulouse vanishing points dataset," in *Proceedings of the 6th ACM Multimedia Systems Conference*. ACM, 2015, pp. 231–236.

[3] M. E. Antone and S. Teller, "Automatic recovery of relative camera rotations for urban scenes," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, vol. 2. IEEE, 2000, pp. 282–289.

[4] C. Arteta, V. Lempitsky, and A. Zisserman, "Counting in the wild," in *European conference on computer vision*. Springer, 2016, pp. 483–498.

[5] O. Barinova, V. Lempitsky, E. Tretiak, and P. Kohli, "Geometric image parsing in man-made environments," in *European conference on computer vision*. Springer, 2010, pp. 57–70.

[6] A. Borji, "Vanishing point detection with convolutional neural networks," *arXiv preprint arXiv:1609.00967*, 2016.

[7] T. Bruls, H. Porav, L. Kunze, and P. Newman, "The right (angled) perspective: Improving the understanding of road scenes using boosted inverse perspective mapping," *arXiv preprint arXiv:1812.00913*, 2018.

[8] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *arXiv preprint arXiv:1405.3531*, 2014.

[9] R. T. Collins and R. S. Weiss, "Vanishing point calculation as a statistical inference on the unit sphere," in *Computer Vision, 1990. Proceedings, Third International Conference on*. IEEE, 1990, pp. 400–403.

[10] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, "Convolutional features for correlation filter based visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 58–66.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. Ieee, 2009, pp. 248–255.

[12] P. Denis, J. H. Elder, and F. J. Estrada, "Efficient edge-based methods for estimating manhattan frames in urban imagery," in *European conference on computer vision*. Springer, 2008, pp. 197–210.

[13] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, " CARLA: An Open Urban Driving Simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[14] M. Dubská, A. Herout, and J. Havel, "PClines–Line detection using parallel coordinates," 2011.

[15] P. Fischer, A. Dosovitskiy, and T. Brox, "Image orientation estimation with convolutional networks," in *German Conference on Pattern Recognition*. Springer, 2015, pp. 368–378.

[16] D. F. Fouhey, W. Hussain, A. Gupta, and M. Hebert, "Single image 3D without a single 3D image," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1053–1061.

[17] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[19] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2980–2988.

[20] L. He, G. Wang, and Z. Hu, "Learning Depth from Single Images with Deep Neural Network Embedding Focal Length," *IEEE Transactions on Image Processing*, 2018.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] J. Lezama, R. Grompone von Gioi, G. Randall, and J.-M. Morel, "Finding vanishing points via point alignments in image primal and dual domains," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 509–515.

[23] M. Liu, X. He, and M. Salzmann, "Geometry-aware Deep Network for Single-Image Novel View Synthesis," *arXiv preprint arXiv:1804.06008*, 2018.

[24] W. Liu, K. Lis, M. Salzmann, and P. Fua, "Geometric and Physical Constraints for Head Plane Crowd Density Estimation in Videos," *arXiv preprint arXiv:1803.08805*, 2018.

[25] W. Liu, M. Salzmann, and P. Fua, "Context-Aware Crowd Counting," *CoRR*, vol. abs/1811.10452, 2018.

[26] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The oxford robotcar dataset," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, 2017.

[27] S. Mahendran, H. Ali, and R. Vidal, "3D pose regression using convolutional neural networks," in *IEEE International Conference on Computer Vision*, vol. 1, no. 2, 2017, p. 4.

[28] R. O'malley, M. Glavin, and E. Jones, "Vision-based detection and tracking of vehicles to the rear with perspective correction in low-light conditions," *IET Intelligent Transport Systems*, vol. 5, no. 1, pp. 1–10, 2011.

[29] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," *arXiv preprint*, 2017.

[30] A. P. Sangmin Oh, Anthony Hoogs *et al.*, "A Large-scale Benchmark Dataset for Event Recognition in Surveillance Video," in *Proceedings of IEEE Comptuer Vision and Pattern Recognition (CVPR)*, 2011.

[31] J. A. Shufelt, "Performance evaluation and analysis of vanishing point detection techniques," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 3, pp. 282–288, 1999.

[32] J. P. Snyder, *Flattening the earth: two thousand years of map projections*. University of Chicago Press, 1997.

[33] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning." in *AAAI*, vol. 4, 2017, p. 12.

[34] T. Tuytelaars, L. Van Gool, M. Proesmans, and T. Moons, "A cascaded hough transform as an aid in aerial image interpretation," in *ICCV*, 1998.

[35] S. Workman, C. Greenwell, M. Zhai, R. Baltenberger, and N. Jacobs, "DEEPFOCAL: a method for direct focal length estimation," in *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1369–1373.

[36] S. Workman, M. Zhai, and N. Jacobs, "Horizon lines in the wild," *arXiv preprint arXiv:1604.02129*, 2016.

[37] M. Zhai, S. Workman, and N. Jacobs, "Detecting vanishing points using global image context in a non-manhattan world," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5657–5665.